Semi-Automatic fitting of deformable 3D models to 2D sketches

by

Xianglong Chang

B.Eng., Huazhong University of Science and Technology, 1994

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES

(Computer Science)

The University of British Columbia

(Vancouver)

April 2008

Abstract

We present a novel method for building 3D models from a user sketch. Given a 2D sketch as input, the approach aligns and deforms a chosen 3D template model to match the sketch. This is guided by a set of user-specified correspondences and an algorithm that deforms the 3D model to match the sketched profile. Our primary contribution is related to fitting the 3D deformable geometry to the 2D user sketch. We demonstrate our technique on several examples.

Table of Contents

List of Figures

Acknowledgements

First of all, I'd like to acknowledge the invaluable supervision from my supervisors Dr. Alla Sheffer and Dr. Michiel van de Panne. Without you, this would have never been possible.

I also would like to thank Sahar Sajadieh for her work on this project during her summer study with Dr. Alla Sheffer. Thanks also to the people from imager lab and my friends in Computer Science Department for the fun, inspiring work and study environment.

Most important, my special thanks to my parents and my wife for encouraging me to complete the study and the greatest support at all time.

XIANGLONG CHANG

The University of British Columbia
April 2008

All my work is dedicated to my family.

Chapter 1

Introduction

1.1 Motivation

Geometry processing techniques are widely used in computer-aided design, computer animation, and related areas. One of the central geometry processing tasks is to create models. Users demand tools to be intuitive, robust, and most importantly, to help in conveying their ideas. Sketching with pen and paper is a traditional way of communicating ideas and shapes. A sketch can convey a shape in a few strokes and thus can be an efficient tool in prototyping and preliminary design. The desire for efficiency and convenience has driven researchers to study sketching as a form of interaction graphical software and develop tools capable of interpreting sketches. Today, a small but increasing number of applications in design, modeling and animation support some form of sketching.

Sketch-based applications in geometry processing area have been around since late 1990's. Creating simple freeform shapes from scratch has been demonstrated in limited form by [11] and [13]. The systems are not able to create complex shapes, because these include a multitude of geometric features, many of which are difficult to infer from a sketch. An alternative is to modify existing templates using sketched feature lines, such as suggestive contours [7]. However, this is by no means trivial. Nealen et al. [20] developed a sketch-based interface to tackle this problem.

The resulting shapes can be realistic and complex, but the approach requires the user to be experienced in manipulating the models. If we could work with existing models while using a sketch as input as in [11] and [13], then the system would be accessible to a wide range of users. Creating a model from a photograph may also become possible.

1.2 System Overview

This thesis presents a method to create new geometric models by fitting 3D deformable templates to 2D sketches. In our system, a user creates a sketch by drawing contour strokes either on a blank canvas or on top of a background image of the object to be modeled. The user also needs to specify a small number of correspondences between the 3D template model and the user drawing. Each correspondence pairs a 2D sketch point with a 3D vertex on the template model. The system uses these correspondences and the associated 2D and 3D normal to align and deform the 3D template so that the resulting model matches the 2D sketch. An example of input and output of the system is shown in Figure 1.1. The corresponding workflow is shown in Figure 1.2.

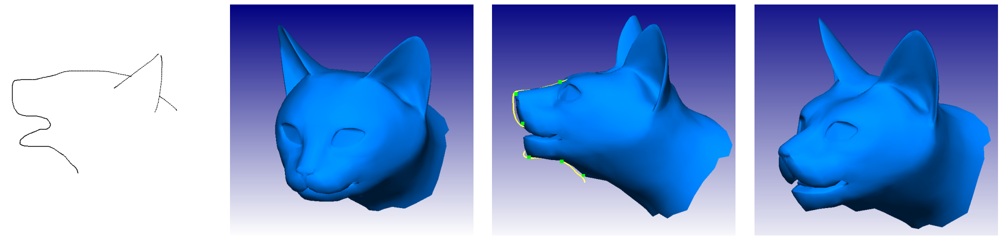

(a) User sketch (b) Template (c) Align and deform (d) Final result

Figure 1.1: The example of input and output of the system

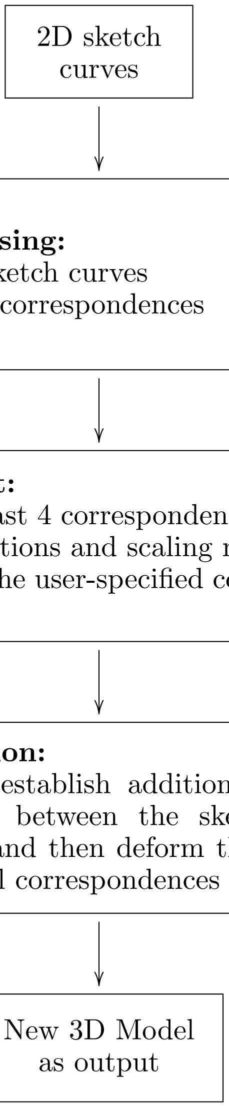
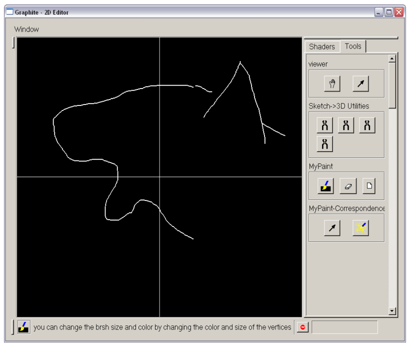

2D sketch curves
Preprocessing:
- Smooth sketch curves
- Establish correspondences
Alignment:
Using at least 4 correspondences, solve for the rotations and scaling needed to best align the user-specified correspondences.
Deformation:
Iteratively establish additional correspondences between the sketch and 3D model and then deform the model based on all correspondences
New 3D Model as output

Figure 1.2: System workflow

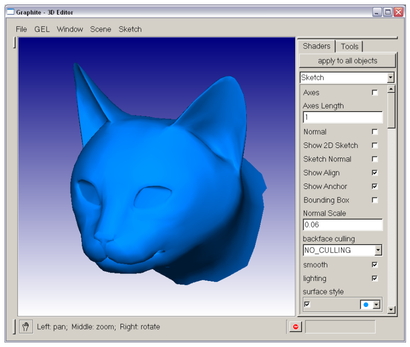
1.2.1 User Interfaces

The system has two separate user interfaces: a 2D sketch interface, and a 3D template editing interface, as shown in Figure 1.3. The sketch consists of strokes, which can be either open or closed. We do require that a unique normal can be inferred for each point on a stroke, this is obtained from knowing the direction of the stroke as it is drawn, as described in more detail in Chapter 3. The user draws with the mouse by holding the left button pressed while dragging the cursor in the sketch area. The user uses both 2D and 3D interfaces in order to specify correspondences between the sketch and the 3D template, one pair a time. Lastly, the 3D interface is used to display the result of the alignment and deformation.



you can change the brsh size and color by changing the color and size of the vertices

(a) 2D user sketch interface



Sketch

Axes ☐

Axes Length: 1

Normal ☐

Show 2D Sketch ☐

Sketch Normal ☐

Show Align ☑

Show Anchor ☑

Bounding Box ☐

Normal Scale: 0.06

backface culling: NO_CULLING

smooth ☑

lighting ☑

surface style ☑

Left: pan; Middle: zoom; Right: rotate

(b) 3D editing interface

Figure 1.3: System user interfaces

1.2.2 Alignment and Deformation

The alignment process transforms the template in order to align the user-specified correspondences as closely as possible. During this step, the shape of the template is preserved and the template only undergoes a similarity transformation. The alignment consists of two steps. First, the system scales the model to the same size as the sketch. Second, it rigidly transforms and translates the model to best achieve the specified correspondences. The output of the alignment is then used as the input to the deformation process.

Once the template is approximately aligned with the sketch, we further deform it so that the deformed model better fits the sketch. Correspondences are first established between the template and the sketch. Then we use both the normal vectors and spatial positions of the corresponding points to determine the appropriate deformations that deform the 3D model towards matching the 2D sketch. Our deformation method is based on Popa et al. [22] where they compute the transformation for each triangle of the mesh by propagating and blending transformations from anchor triangles, and use vertex positions before and after the deformation to represent the transformation gradients. The optimal vertex positions can then be obtained by minimizing the difference between the transformation gradients and the previously computed transformations. The original method uses only normal constraints. In our application, we also consider positional constraints in order to obtain a better match. Therefore, we modify the optimization scheme of the original method by introducing an extra term in the optimization metric to account for the positional constraints. We further introduce additional constants that balance the relative contributions of the desire to maintain the original template shape and positional constraints. The modified method is described in Chapter 4.

1.3 Organization

The rest of this thesis is organized as follows. Relevant literature is discussed in Chapter 2. The 3D to 2D alignment is described in Chapter 3. Mesh deformations is discussed in Chapter 4, and Chapter 5 demonstrates and explains the results. Lastly Chapter 6 discusses possible improvements and as well as future research.

Chapter 2

Related Work

Our proposed modeling method involves aspects of sketch-based modeling and reconstruction, registration, alignment, and shape deformation. In this chapter we review previous work in these areas.

2.1 Sketch-based Modeling and Reconstruction

Sketch-based modeling potentially provides users with a convenient way to interactively create new objects on a computer either from scratch or by deforming existing templates. It combines a user interface with mesh editing methods.

2.1.1 Freeform-based Modeling

Igarashi et al. [11] designed a sketch-based interface called Teddy to generate 3D freeform models. The user draws several 2D freeform strokes representing a desired geometry, and the system automatically generates a plausible 3D model by inflating [30] the region surrounded by the user sketch. The creation process requires silhouettes to be simple and closed curves and the geometries it creates are topologically equivalent to a sphere.

Karpenko and Hughes [13] present a method which can be incorporated into a sketch-based freeform modeling interface, such as Teddy [11], to extend the ability of creating more complex freeform geometries. The algorithm is based on the work of [24] and [31] to infer a plausible shape from its visible contours. The sketch is no longer required to be a simple closed curve, but can have cusps and T-junctions. To create a desired 3D geometry, they use a mass-spring system to connect the topological embedding and smoothen the raw solid shape to obtain the final geometry.

2.1.2 Template-based Modeling

Nealen et al. [20] present a view-dependent sketch-based interface for mesh editing. They adopted the Laplacian framework to deform the shape of the template to get a desired model. They suggest that building a model from sketch can be seen as an inverse Non-Photorealistic-Rendering problem, and therefore their interface uses silhouettes and suggestive contours as input for the editing mode. Users sketch a curve within a user specified region of influence on the model and the system adapts the template so that the curve becomes a feature line on the model.

Kho and Garland [14] present a system that allows users to interactively modify a template via direct sketching on the model. They treat the user sketch as a reference skeleton of the region of the interest(ROI). Users first draw a reference curve to define a skeleton of the ROI, then draw another target reference curve as the skeleton of the desired shape. The deformation of the template is determined by the deformation of the reference curves.

Yang et al. [33] developed a sketch-based system for modeling of parameterized objects. Their system creates 3D geometries of particular object classes from 2D user sketches. It interprets the strokes as a sketch graph and uses the 2D template to label the parts of the sketch. Once a best-fit template is found, the system constructs a new model by extracting measurements from the resulting labeled sketch. These measurements are fed to a pre-existing parameterized 3D model generator for each object class. They demonstrate results using planes, cups, and fish (Figure 2.1). The method can be extended to work with other parameterizable

3D objects.

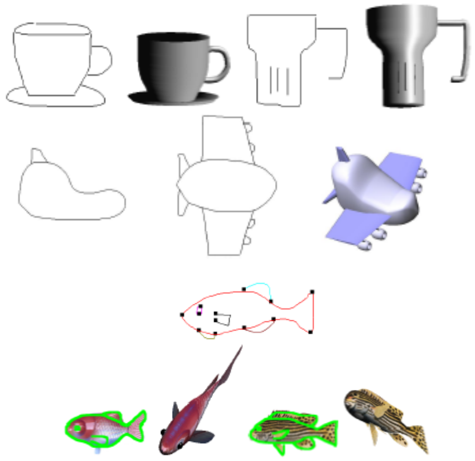


Figure 2.1: Sketch-based modeling of parameterized objects [33]

2.2 Registration and Alignment

Registration and alignment are required in our work to find the relationship between the template model and the sketch. Many applications of surface matching or shape recognition also have similar requirements. Normally, registrations can be used to align scanned point-based surfaces to 3D models [17], align 2D textures to a 3D model [16], align 3D models with respect to one another, etc.

Besl and McKay [5] present an iterative closest point (*ICP*) algorithm for registration of 3D shapes. The method is independent of the representations of the geometric data and iteratively solves the registration problem of 3D shapes. During each iteration, it finds the closest point on a geometric model to a given point; then a rigid transformation is computed by minimizing the distances between the transformed points of the given points and their corresponding points on the model. The iteration is repeated using the updated points from previous iteration steps and stops once the error metric falls below a threshold. The convergence of the algorithm depends on a good initial estimate of the registration.

Li et al. [17] present a feature-based algorithm for approximate scan-to-model alignment. Their method estimates the transformation that matches two surfaces from a single pair of corresponding points with their surface positions, normals and principle curvature directions. However, they need to do an exhaustive search over all possible correspondences from scan data and the model data to find an optimal pair of corresponding points. The result can then be used as the input to an *ICP* algorithm to refine the registration between the surface and the model.

2.3 Shape Deformations

2.3.1 Deformations Based on Gradients

Mesh editing and deformation are key topics in geometric modeling. Yu et al. [34] use a Poisson equation to edit meshes through gradient field manipulation. The method relies on both the guidance vector field, i.e. gradient fields, and boundary conditions. They used geodesic distance between every free vertex and constrained vertices to evaluate the guidance vector field. Manipulations to the guidance field yield local transformations defined on a per triangle basis, the system then propagates the local transformations to create a smooth transition.

Sumner et al. [29] propose a deformation gradient based method for transferring large deformations from source to target models. They use the correspondences between the source and target triangles to define how the deformation of the source mesh should be transferred to the target. Once the correspondences are found, the system evaluates the transformations induced by the deformation of the source model and then maps the transformations through the correspondences from the source to the target. Finally, the system solves an optimization problem to enforce the continuous shape requirements of the target mesh. Popa et al. [22] also use deformation gradients in their method. They add material properties to the model to allow for non-uniform transformations as shown in Figure 2.2.

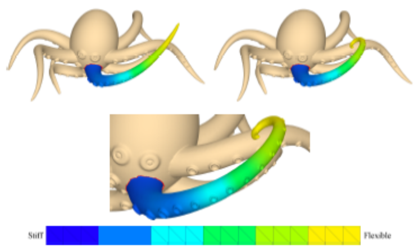


Figure 2.2: Material-aware mesh deformations [22]

The reconstruction of meshes with desired gradients is an overdetermined problem, and is thus solved in a weighted least-squares sense. These methods work well for rotations but not for translations, as the gradient field is not sensitive to translation. Deformations containing both rotation and translations therefore remain problematic.

2.3.2 Deformations Based on Differential Coordinates

Another class of methods modifies the differential coordinates of the surface vertices. Alexa [1] demonstrates that using Laplacian coordinates as a mesh representation in morphing is better than using traditional Cartesian coordinates, and proposes to use varying weights during linear interpolation to obtain more natural, gradual blending. The Laplacian coordinate of a vertex v_i is represented by the difference between v_i and the average of its neighbors:

$$\delta_i = v_i - \frac{1}{n_i} \sum_{j \in Nbr(v_i)} v_j \tag{2.1}$$

where $Nbr(v_i)$ is the set of the immediate neighboring vertices of the vertex v_i and $n_i = |Nbr(v_i)|$ or the valence of vertex v_i.

There are a number of methods that use variations of the Laplacians coordinates for mesh deformation [18, 19, 28, 27]. Sorkine [27] provides a review of current research in geometric processing that is related to Laplacian processing framework and differential representations. Lipman et al. [18] use Laplacian Coordinates to preserve the high frequency of mesh surfaces where they solve the rotation invariant problem by first estimating rotations of a local frame and then rotating original Lalacians with these estimated rotations. Sorkine et al. [28] develop a method to implicitly solve for rotations of Laplacians by finding optimal transformations for each vertex. They linearize the local frame transformations to make computations more efficient, but at the cost of unavoidable artifacts for large rotations.

Sheffer and Kraevoy [25] present a method that uses pyramid coordinates for 3D mesh morphing and deformation. Pyramid coordinates are a local shape representation based on a set of angles and the edge lengths relating a vertex to its adjacent neighbors. They are invariant under rigid transformations. For mesh deformation, their method uses a number of user-specified control vertices to evaluate deformations consisting of scaling, bending and rotation. The method is robust and capable of generating natural looking deformations and morphing sequences (see Figure 2.3). For reconstruction, they project the vertex v and its neighboring vertices v_i onto v's projection plane. Given v', the projection of v, and the information of the offsets of all v_i's, the new position of the vertex v can be reconstructed.

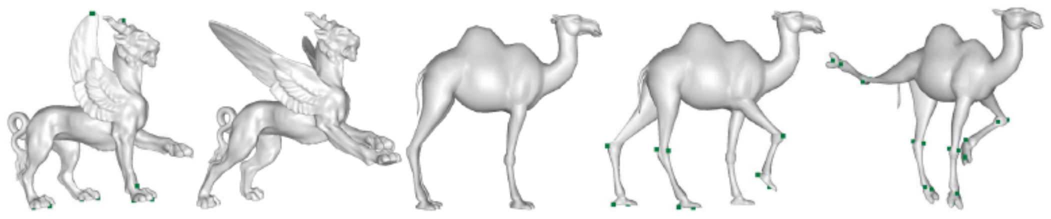


Figure 2.3: Pyramid coordinates for morphing and deformation [25]

The above methods do not take the volume change into consideration when deforming a mesh surface. Zhou et al. [35] present a method to process large deformations using the volumetric graph Laplacian to avoid unnatural volume change and local self-intersection by building two lattices wrapping around the mesh from both inside and outside. The internal graph fills the interior volume to prevent large volume change, and the external graph prevents local self-intersection. This hierarchy can better preserve both the local details and volumes, but also significantly increases the complexity of the operations.

Chapter 3

3D to 2D Alignment

3.1 2D Sketch

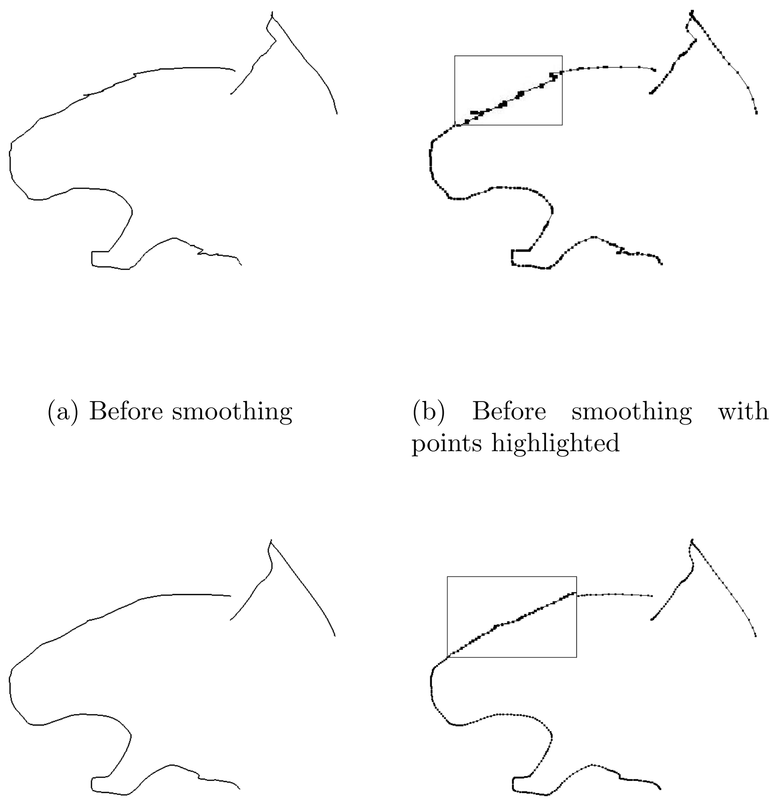
In our system, we use the 2D sketch to infer an intended 3D shape. To serve our purpose, the points on any sketched curve are required to have unique 2D normals. We infer the normal direction from the direction of the stroke using the counter-clockwise convection. The resulting normal vector of each sample point will be assumed to point outwards with respect to the contour direction.

3.1.1 Smoothing

The hand drawn sketch is prone to noise. Smoothness is important to our system because the normal vectors of sketch points are used as the input for mesh deformation. As a result, we apply the following Laplacian smoothing formula to all the sketch points

$$p'_i = \lambda \left(\frac{1}{n} \sum_{j=1}^{n} q_j \right) + (1 - \lambda)\, p_i \tag{3.1}$$

where p'_i is the smoothed position of p_i, λ is the coefficient to balance the contributions between p_i and its neighbors, n is the number of points in the neighborhood of p_i denoted as $Nbr(p_i)$, and $q_j \in Nbr(p_i)$.



(a) Before smoothing

(b) Before smoothing with points highlighted

(c) After smoothing

(d) After smoothing with points highlighted

Figure 3.1: Smoothing of the sketch

3.2 Correspondence

Correspondences between the sketch and the model are supplied by the user. The user selects a point from the sketch and a corresponding vertex from the model establishing a corresponding pair. When selecting the 3D corresponding vertex, we also determine one of its surrounding facets as the *anchor facet*. We use the same definition of the anchor facet as in [22]. The anchor facets are needed to determine the deformation gradients for mesh deformations, as will be explained in detail in Chapter 4. The user repeats the selection process, one pair a time, until there are enough correspondences for our operations. We require a minimum set of four correspondences for alignment. Additional correspondences can serve to help with mesh deformations.

In our implementation, we use Graphite's [8] built-in function to select the vertices of the 3D model as the 3D points for correspondences. For the 2D sketch, we use the mouse to select a point on the 2D strokes and the system then locates the point closest to the selected location as illustrated in Figure 3.2.

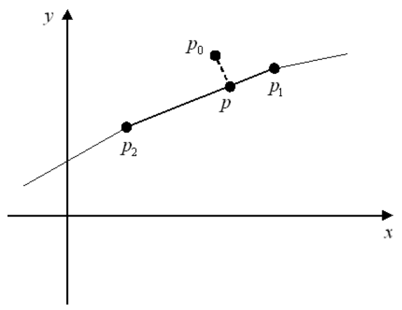


Figure 3.2: Assuming that p_1 and p_2 are two adjacent sketch points, p_0 is the point that user clicked on the screen, then the system will locate point p on the stroke and return it as a corresponding 2D point.

Once the correspondences are established, the spatial coordinates of the sketch points are then used as position constraints for both alignment and mesh deformation.

3.3 Alignment

We wish to find a similarity transformation that aligns the template and the sketch. For a generic rigid transformation matrix M as shown in a homogeneous form in equation 3.2, there are 12 unknowns of which 9 unknowns, r_{ij}, $i, j \in [0, 2]$, are for the rotation or reflection, and 3 unknowns, t_x, t_y, t_z, are for the translation. In our system we have to deal with uniform scaling, rotations and translations.

$$M = \begin{bmatrix} r_{00} & r_{01} & r_{02} & t_x \\ r_{10} & r_{11} & r_{12} & t_y \\ r_{20} & r_{21} & r_{22} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

However, we are not able to unambiguously determine the 3D transformation matrix required for alignment in one step. Therefore, we developed a procedure to do alignment in three steps.

Given a set of 2D correspondences supplied by the user as described in Section 3.2, we model the transformation matrix M by separating the overall transformation into three groups of transformations. Each of the three successive transformations contains scaling, rotation and translation. However, the rotation is fixed to rotate around a specific axis of the Cartesian coordinate system, i.e. the rotation axis is either X, Y, or Z. It can be defined mathematically as:

$$v'_i = M(v_i) = M_x(M_y(M_z(v_i))) \tag{3.3}$$

where v_i is the i^{th} vertex of the original 3D template, v'_i is the i^{th} vertex of the aligned model, and for each set of transformations:

$$M_k(v) = R_k(s_k v) + T_k \tag{3.4}$$

M_k is a group of transformations and $k = x$, y, or z. R_k is a rotation matrix, s_k is a scale factor, and T_k is a translation vector. The alignment problem can then be factored into the subproblems of scaling, rotation and translation.

3.3.1 Scaling

Given a set of correspondences, we can estimate a scale factor s by solving the following optimization problem (3.5)

$$\min_s \sum_{\forall(i,j)} \left(s \left\| p_i - p_j \right\|_2^2 - \left\| p'_i - p'_j \right\|_2^2 \right)^2 \tag{3.5}$$

where p_i, p_j are 3D vertices, and p'_i, p'_j are corresponding 2D sketch points.

3.3.2 Rotation and Translation

As shown in equation (3.3), we need to find up to three sets of transformations with respect to the rotation axis, and we solve the following optimization metric for a specific rotation matrix R and translation vector T.

$$\min \sum_{i}^{n} \|M(v_i) - p_i\|_2^2 \tag{3.6}$$

where n is the number of the correspondences and we require $n \geq 4$, v_i is a vertex of the 3D model and p_i is the corresponding 2D sketch point. Our problem is under-constrained in that p_i is only a 2D point with which we are not able to fully constrain the position of a 3D vertex during the transformation. Therefore, we make an assumption that a 2D sketch is drawn on a 3D plane. In our implementation, we use the XZ plane, and set the Y-component of the translation vector to zero which means we accept whatever value of y-coordinate of the transformed vertex v_i after a transformation. We will then be able to proceed solving the optimization problem (3.6) for a rotation matrix and a translation vector. As shown in equation (3.4), we have to solve for three degrees of freedom for alignment.

Transformations with Rotation around X-Axis

A 3D model rotation about the X-Axis can be expressed using the rotation matrix :

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C & -S \\ 0 & S & C \end{bmatrix}$$

where $C = \cos\theta$, $S = \sin\theta$, and θ is the rotation angle. Therefore, the optimization problem (3.6) can be written explicitly as:

$$\min_{C,S,t_x,t_z} \sum_i^n \left\| \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & C & -S & 0 \\ 0 & S & C & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x\, x_i \\ s_x\, y_i \\ s_x\, z_i \\ 1 \end{bmatrix} - \begin{bmatrix} x'_i \\ 0 \\ z'_i \\ 1 \end{bmatrix} \right\|_2^2 \tag{3.7}$$

where s_x is the scale factor computed as described in Section 3.3.1, $t = [t_x,\ 0,\ t_z]^T$, x_i, y_i, z_i are the coordinates of a vertex v_i of the 3D model and x'_i, z'_i are the coordinates of the corresponding point p_i on the 2D sketch. After algebraic simplifications of (3.7), we obtain

$$\begin{aligned} &\min_{C,S,t_x,t_z} \sum_i^n \left[\left(Sy_i^s + Cz_i^s + t_z - z'_i\right)^2 + \left(x_i^s + t_x - x'_i\right)^2 \right] \\ &\text{Subject to}: \ C^2 + S^2 = 1 \end{aligned} \tag{3.8}$$

where $x_i^s = s_x x_i$, $y_i^s = s_x y_i$, $z_i^s = s_x z_i$. As each of the two terms in (3.8) must be positive, and involves an independent set of variables, the entire expression is minimized when both of them are minimized. Hence, we can first solve the second term for t_x

$$\min_{t_x} \sum_i^n \left(x_i^s + t_x - x'_i\right)^2 \tag{3.9}$$

Taking derivative w.r.t t_x, and setting it equal to 0, we obtain the following solution

$$t_x = \frac{1}{n} \sum_i^n \left(x'_i - x_i^s\right) \tag{3.10}$$

Now we still need to solve the first term for C, S, t_z, namely:

$$\begin{aligned} &\min_{C,S,t_z} \sum_i^n \left(Sy_i^s + Cz_i^s + t_z - z'_i\right)^2 \\ &\text{Subject to}: \ C^2 + S^2 = 1 \end{aligned} \tag{3.11}$$

This can be treated as a minimization problem with an equality constraint, and we shall use a penalty method to solve it as will be explained in more detail in section

3.4. The optimization steps for the two remaining transformations are determined in an analogous fashion.

Transformations with Rotation around Y-Axis

The rotation matrix R is as follows:

$$R = \begin{bmatrix} C & 0 & S \\ 0 & 1 & 0 \\ -S & 0 & C \end{bmatrix}$$

Equation (3.6) therefore becomes

$$\min_{C,S,t_x,t_y,t_z} \sum_i^n \left\| \begin{bmatrix} C & 0 & S & t_x \\ 0 & 1 & 0 & 0 \\ -S & 0 & C & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_y\, x_i \\ s_y\, y_i \\ s_y\, z_i \\ 1 \end{bmatrix} - \begin{bmatrix} x'_i \\ 0 \\ z'_i \\ 1 \end{bmatrix} \right\|_2^2 \tag{3.12}$$

With a similar derivation, we can obtain an equivalent form of the scheme (3.12):

$$\min_{C,S,t_x,t_z} \sum_i^n \left[\left(Cx_i^s + Sz_i^s + t_x - x'_i\right)^2 + \left(-Sx_i^s + Cz_i^s + t_z - z'_i\right)^2 \right] \tag{3.13}$$
$$\text{Subject to : } C^2 + S^2 = 1$$

where s_y is the scale factor and $x_i^s = s_y x_i$, $y_i^s = s_y y_i$, $z_i^s = s_y z_i$.

Transformations with Rotation around Z-Axis

The rotation matrix R becomes:

$$R = \begin{bmatrix} C & -S & 0 \\ S & C & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Equation (3.6) is then:

$$\min_{C,S,t_x,t_z} \sum_i^n \left\| \begin{bmatrix} C & -S & 0 & t_x \\ S & C & 0 & 0 \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_z\, x_i \\ s_z\, y_i \\ s_z\, z_i \\ 1 \end{bmatrix} - \begin{bmatrix} x_i' \\ 0 \\ z_i' \\ 1 \end{bmatrix} \right\|_2^2 \tag{3.14}$$

which is equivalent to

$$\min_{C,S,t_x,t_z} \sum_i^n \left[\left(Cx_i^s - Sy_i^s + t_x - x_i'\right)^2 + \left(z_i^s + t_z - z_i'\right)^2 \right] \tag{3.15}$$
$$\text{Subject to}: \; C^2 + S^2 = 1$$

We can solve for

$$t_z = \frac{1}{n} \sum_i^n \left(z_i' - z_i^s\right) \tag{3.16}$$

and then minimize the following optimization scheme for C, S, and t_x.

$$\min_{C,S,t_x} \sum_i^n \left(Cx_i^s - Sy_i^s + t_x - x_i'\right)^2 \tag{3.17}$$
$$\text{Subject to}: \; C^2 + S^2 = 1$$

where s_z is the scale factor and $x_i^s = s_z x_i$, $y_i^s = s_z y_i$, $z_i^s = s_z z_i$.

3.4 Optimization

In this section, we briefly discuss using penalty method [21] to solve our optimization schemes (3.11), (3.13), and (3.17).

3.4.1 Penalty Method

In general, the penalty method can be considered when solving an optimization problem with equality constraints:

$$\min \;\; f(\mathbf{x}) \tag{3.18}$$
$$s.t. \;\; c_i(\mathbf{x}) = 0, \;\; i = 1, 2, \ldots, m$$

The optimization problem is cast as an unconstrained problem by creating a new objective function $\phi\left(\mathbf{x};\mu\right)$

$$\min\ \phi\left(\mathbf{x};\mu\right)=f\left(\mathbf{x}\right)+\frac{1}{2\mu}\sum_{i=1}^{m}c_i^2\left(\mathbf{x}\right) \tag{3.19}$$

Similarly, the first order condition requires:

$$\nabla\phi\left(\mathbf{x};\mu\right)=\nabla f\left(\mathbf{x}\right)+\sum_{i=1}^{m}\frac{c_i\left(\mathbf{x}\right)}{\mu}\nabla c_i\left(\mathbf{x}\right)=0 \tag{3.20}$$

For a sequence of values of μ. We can use the solution $\mathbf{x}^*$ of the $(k-1)$'s unconstrained problem (3.19) with $\mu=\mu_k$. This is a continuation technique. The penalty method algorithm is shown in Algorithm (1) and we choose the appropriate μ during iterations by the following criteria

$$\mu_{k+1}=\begin{cases}0.7\mu_k & \text{if } \phi(\mathbf{x},\mu_k) \text{ was difficult to solve}\\ 0.1\mu_k & \text{if } \phi(\mathbf{x},\mu_k) \text{ was easy to solve}\end{cases}$$

Algorithm 1 Quadratic penalty method

```
Input: Given $\mu_0 > 0$, $\mathbf{x}_0$ and a final tolerance $tol$
Output: $\mathbf{x}_k^*$
 1: $\mathbf{x} = \mathbf{x}_0$
 2: $\mu = \mu_0$
 3: for $k = 0, 1, 2, \ldots$ do
 4:   if $\|\nabla\phi\left(\mathbf{x};\mu\right)\| \leq tol$ then
 5:     $\mathbf{x}_k^* = \mathbf{x}$
 6:     Exit
 7:   else
 8:     choose $\mu_{k+1} \in (0, \mu_k)$
 9:     Use Newton's method with line search to get $\mathbf{x}_{k+1}$, see Algorithm (2).
10:     $\mathbf{x} = \mathbf{x}_{k+1}$
11:     $\mu = \mu_{k+1}$
12:   end if
13: end for
```

3.4.2 Derivation

We have already given three independent optimization schemes for rotation and translation with respect to each coordinate axis. When solving them using penalty

Algorithm 2 Newton's method with line search

Input: $f(\mathbf{x}), \mathbf{x}_0$, maxiterations, α_{min}

Output: $\mathbf{x}^*$

```
 1: Give an initial guess: $\mathbf{x} = \mathbf{x}_0$ and $k = 0$
 2: while $k <$ maxiterations do
 3:   First try Newton's method with $\mathbf{x}_k$
 4:   if The error is within the tolerance then
 5:     Accept the result from Newton's method and return $\mathbf{x}_k$ as $\mathbf{x}^*$
 6:   else
 7:     Try a line search to get a new $\mathbf{x}_k$ and $\alpha$
 8:     if $\alpha < \alpha_{min}$ then
 9:       line search fails and if we have not reached the max number of iterations,
          then try it again. Otherwise, we return the best solution of this search.
10:     else
11:       return $\mathbf{x}_k$ as $\mathbf{x}^*$
12:     end if
13:   end if
14: end while
```

method, we need to reformulate the original constrained optimization to unconstrained optimization as explained in section (3.4.1). The remainder of this section shows the derivation of the corresponding gradient vectors and Hessian matrices.

Transformations with Rotation around X-Axis

The new optimization is obtained from expanding scheme (3.11)as follows:

$$\min_{C,S,t_z} \sum_i^n \left(Sy_i + Cz_i + t_z - z_i'\right)^2 + \frac{1}{2\mu}\left(C^2 + S^2 - 1\right)^2 \tag{3.21}$$

let

$$\phi(\mathbf{x}, \mu) = \sum_i^n \left(Sy_i + Cz_i + t_z - z_i'\right)^2 + \frac{1}{2\mu}\left(C^2 + S^2 - 1\right)^2 \tag{3.22}$$

$\mathbf{x} = [C,\ S,\ t_z]^T$, μ is the penalty coefficient. Hence, the gradient and Hessian of $\phi(\mathbf{x}, \mu)$ are

$$\nabla\phi(\mathbf{x}, \mu) = \begin{bmatrix} \sum_i^n 2z_i\left(Cz_i + Sy_i + t_z - z_i'\right) + \frac{2C\left(C^2+S^2-1\right)}{\mu} \\ \sum_i^n 2y_i\left(Cz_i + Sy_i + t_z - z_i'\right) + \frac{2S\left(C^2+S^2-1\right)}{\mu} \\ \sum_i^n 2\left(Cz_i + Sy_i + t_z - z_i'\right) \end{bmatrix} \tag{3.23}$$

and

$$\textbf{Hessian}_\phi = \begin{bmatrix} \sum_i^n 2z_i^2 + \frac{6C^2+2S^2-2}{\mu} & (\sum_i^n 2y_i z_i) + \frac{4CS}{\mu} & \sum_i^n 2z_i \\ (\sum_i^n 2y_i z_i) + \frac{4CS}{\mu} & \sum_i^n 2y_i^2 + \frac{2C^2+6S^2-2}{\mu} & \sum_i^n 2y_i \\ \sum_i^n 2z_i & \sum_i^n 2y_i & \sum_i^n 2 \end{bmatrix} \tag{3.24}$$

Transformations with Rotation around Y-Axis

We expand scheme (3.13) as:

$$\begin{aligned} \min_{C,S,t_x,t_z} \quad & \sum_i^n \left[\left(Cx_i + Sz_i + t_x - x_i'\right)^2 + \left(-Sx_i + Cz_i + t_z - z_i'\right)^2 \right] \\ & + \frac{\left(C^2+S^2-1\right)^2}{2\mu} \end{aligned} \tag{3.25}$$

and similarly,

$$\begin{aligned} \phi(\mathbf{x},\mu) &= \sum_i^n \left[\left(Cx_i + Sz_i + t_x - x_i'\right)^2 + \left(-Sx_i + Cz_i + t_z - z_i'\right)^2 \right] \\ &+ \frac{\left(C^2+S^2-1\right)^2}{2\mu} \end{aligned} \tag{3.26}$$

where $\mathbf{x} = [C,\ S,\ t_x,\ t_z]^T$ and μ is the same as before. Hence, the gradient$\nabla\phi(\mathbf{x},\mu)$

$$\begin{bmatrix} \sum_i^n 2x_i\left(Cx_i + Sz_i + t_x - x_i'\right) + 2z_i\left(-Sx_i + Cz_i + t_z - z_i'\right) + \frac{2C\left(C^2+S^2-1\right)}{\mu} \\ \sum_i^n -2z_i\left(Cx_i + Sz_i + t_x - x_i'\right) + 2x_i\left(-Sx_i + Cz_i + t_z - z_i'\right) + \frac{2S\left(C^2+S^2-1\right)}{\mu} \\ \sum_i^n 2\left(Cx_i + Sz_i + t_x - x_i'\right) \\ \sum_i^n 2\left(-Sx_i + Cz_i + t_z - z_i'\right) \end{bmatrix}$$

and the Hessian of $\phi(\mathbf{x},\mu)$

$$\begin{bmatrix} \sum_i^n 2\left(x_i^2 + z_i^2\right) + \frac{6C^2+2S^2-2}{\mu} & \frac{4CS}{\mu} & \sum_i^n 2x_i & \sum_i^n 2z_i \\ \frac{4CS}{\mu} & \sum_i^n 2\left(x_i^2 + z_i^2\right) + \frac{2C^2+6S^2-2}{\mu} & \sum_i^n 2z_i & -\sum_i^n 2x_i \\ \sum_i^n 2x_i & \sum_i^n 2z_i & \sum_i^n 2 & 0 \\ \sum_i^n 2z_i & -\sum_i^n 2x_i & 0 & \sum_i^n 2 \end{bmatrix}$$

Transformations with Rotation around Z-Axis

Expand the scheme (3.17):

$$\min_{C,S,t_x} \sum_i^n \left(Cx_i - Sy_i + t_x - x_i'\right)^2 + \frac{\left(C^2 + S^2 - 1\right)^2}{2\mu} \tag{3.27}$$

and

$$\phi(\mathbf{x}, \mu) = \sum_i^n \left(Cx_i - Sy_i + t_x - x_i'\right)^2 + \frac{\left(C^2 + S^2 - 1\right)^2}{2\mu} \tag{3.28}$$

where $\mathbf{x} = [C,\ S,\ t_z]^T$ and μ is the same as before. Hence, the gradient and Hessian of $\phi(\mathbf{x}, \mu)$ are

$$\nabla\phi(\mathbf{x}, \mu) = \begin{bmatrix} \sum_i^n 2x_i \left(Cx_i - Sy_i + t_x - x_i'\right) + \frac{2C\left(C^2+S^2-1\right)}{\mu} \\ \sum_i^n -2y_i \left(Cx_i - Sy_i + t_x - x_i'\right) + \frac{2S\left(C^2+S^2-1\right)}{\mu} \\ \sum_i^n 2 \left(Cx_i - Sy_i + t_x - x_i'\right) \end{bmatrix} \tag{3.29}$$

and

$$\mathbf{Hessian}_\phi = \begin{bmatrix} \sum_i^n 2x_i^2 + \frac{6C^2+2S^2-2}{\mu} & -\sum_i^n 2x_iy_i + \frac{4CS}{\mu} & \sum_i^n 2x_i \\ -\sum_i^n 2x_iy_i + \frac{4CS}{\mu} & \sum_i^n 2y_i^2 + \frac{2C^2+6S^2-2}{\mu} & -\sum_i^n 2y_i \\ \sum_i^n 2x_i & -\sum_i^n 2y_i & \sum_i^n 2 \end{bmatrix} \tag{3.30}$$

Chapter 4

Mesh Deformations

After the alignment step, the template will be overlapping the 2D sketch, but the overall shape of the template remains unchanged since the alignment is a rigid transformation. Hence, we now deform the aligned template to better match the sketch. Our mesh deformation method is based on the method given by Popa et al. [22], and we modify it to incorporate the positional constraints.

4.1 Method Overview

4.1.1 Optimization Scheme

In [22], a 3D model is deformed by utilizing a set of user specified anchors. First, it computes the anchor transformation matrices using the rotation angle and axis which are given by users when selecting anchors, and then propagates the transformations to the entire model by blending them with each facet of the 3D model. Finally, it finds an optimal position for each vertex of the mesh by minimizing the errors between previously computed transformations and the transformations obtained from the transformation gradient. The optimization scheme is formally modeled as:

$$\min_{\tilde{v}} \sum_{i}^{n-q} \psi_i \| \widetilde{V}_i V_i^{-1} - T_i \|_F^2 \tag{4.1}$$

where n is the number of faces, q is the number of the anchors, ψ_i is the shearing stiffness coefficient, T_i is the transformation for i^{th} triangle of the mesh, V_i and $\widetilde{V_i}$ are the local frames of the i^{th} triangle before and after applying the deformation and defined as follows:

$$V_i = (v_4 - v_1, v_4 - v_2, v_4 - v_3) \tag{4.2}$$

where v_1, v_2, and v_3 are the three vertices of the i^{th} triangle, v_4 is the augmented fourth vertex computed by offsetting one of the vertices by the triangle normal, such that three vectors $(v_4 - v_1, v_4 - v_2, v_4 - v_3)$ define a local frame. After applying the deformation, V_i becomes $\widetilde{V_i}$

$$\widetilde{V_i} = (\tilde{v_4} - \tilde{v_1}, \tilde{v_4} - \tilde{v_2}, \tilde{v_4} - \tilde{v_3}) \tag{4.3}$$

We refer readers to [22] for a detailed explanation.

The anchor triangles in the system only provide information for the rotations, and therefore the position constraints are not included in the above formulation. In our system, however, a good match between the sketch and model requires the correspondences to be as close as possible after deformation, thus we must enforce the position constraints. We modify the optimization scheme given in (4.1) as follows:

$$\min_{\tilde{v}} \sum_{i}^{n} \alpha \left(\psi_i \|\widetilde{V_i} V_i^{-1} - T_i\|_F^2 \right) + \sum_{j}^{m} \beta \ \|\tilde{v_j} - c_j\|_2^2 \tag{4.4}$$

where α and β are weighting factors to balance the contributions between the transformation and position constraints, n is the number of faces and m is the number of correspondences; $\tilde{v_j}$ is the deformed vertex of v_j and c_j is the corresponding 2D sketch point of v_j. Note that when we compute the error between $\tilde{v_j}$ and c_j, we only consider the X and Z components of $\tilde{v_j}$, since our sketch is assumed to be on the XZ plane. We thus change the framework of [22] to accommodate 2D sketch points as position constraints. We do not have restrictions on choosing the values of α and β. In our experiments, we use various values for them, but make $\beta > \alpha$

so that position constraints are given significant weight. Solving this optimization problem yields new positions of the vertices such that the correspondences are as close as possible.

4.1.2 Constraints

As described above, we shall use both the normal vectors and the spatial positions of the correspondences as constraints. The spatial positions are taken from the corresponding points. We use the normal vectors of the anchor facets as the normal vectors of the 3D correspondences. For the sketch, we modify the 2D normal vectors of the sketch points as described below so that we are able to use them to evaluate the deformations.

For a correspondence pair (v_j, c_j), where v_j is a vertex of the template, and c_j is a 2D sketch point, assume that $\overrightarrow{N_c} = [n_x^c, n_z^c]^T$ is the normal vector of c_j, and $\overrightarrow{N_v} = [n_x^v, n_y^v, n_z^v]^T$ is the normal vector of the anchor facet which contains v_j. Then we modify $\overrightarrow{N_c}$ to obtain $\overrightarrow{N_c}' = [n_x^c, n_y^v, n_z^c]^T$. Note that we do not take 0 as the y-component of $\overrightarrow{N_c}'$, even though we have assumed that the 2D sketch is on XZ coordinate plane. When using a 2D vector to infer a 3D vector, we do not have enough information on what a best fit y-component should be. Therefore, we consider only the differences between $\overrightarrow{N_v}$ and $\overrightarrow{N_c}$ in the X and Z directions. Experiments have also shown that using 0 valued y-component results in larger errors.

4.2 Deformation

In order to evaluate the appropriate deformation of the mesh, we need to estimate the transformation of each triangle on the mesh. As given in [22], the transformations of individual triangles can be obtained by blending the transformations of the anchor facets. The transformations of anchor facets can be evaluated as described below.

Assuming that the normal vector of an anchor triangle f_a is $\overrightarrow{N}_{model} =$

$[x_m,\ y_m,\ z_m]^T$ and its corresponding augmented normal vector from the 2D sketch(see section 4.1.2) is $\overrightarrow{N}_{sketch} = [x_s,\ y_s,\ z_s]^T$. Then

$$\begin{aligned} \theta &= \arccos\left(\overrightarrow{N}_{sketch} \cdot \overrightarrow{N}_{model}\right) \\ \overrightarrow{w} &= \overrightarrow{N}_{sketch} \times \overrightarrow{N}_{model} \end{aligned} \tag{4.5}$$

where θ is the rotation angle and $\overrightarrow{w}$ is the rotation axis, therefore, the transformation matrix is

$$T_a = \begin{bmatrix} tx_w x_w + \cos\theta & tx_w y_w + z_w \sin\theta & tx_w z_w - y_w \sin\theta \\ tx_w y_w - z_w \sin\theta & ty_w y_w + \cos\theta & ty_w z_w + x_w \sin\theta \\ tx_w z_w + y_w \sin\theta & ty_w z_w - x_w \sin\theta & tz_w z_w + \cos\theta \end{bmatrix} \tag{4.6}$$

where $t = 1 - \cos\theta$ and $x_w,\ y_w,\ z_w$ are the three components of $\overrightarrow{w}$. We find the transformation for each anchor facet by using (4.6) and then compute T_i as follows:

$$T_i = \bigoplus_{a=1}^{q} \omega_i^a \odot T_a \tag{4.7}$$

where $i = 1, \dots, n$, T_a is the transformation of an anchor triangle, $a = 1, \dots, q$, and ω_i^a represents the relative influence of the anchor transformation T_a to T_i. The details of weights computation for each triangle of the mesh are given in [22]. Finally, we solve the optimization problem (4.4) with $\{T_1,\ \dots,\ T_n\}$ to compute the newly deformed shape.

4.3 Numerical Methods

The optimization formulation given by (4.1) can be rewritten as a linear system:

$$\min_{\tilde{v}} \|\Psi\left(A\tilde{v} - t\right)\|_2^2 \tag{4.8}$$

where A is a sparse matrix constructed using $\widetilde{V}_i$ and V_i, t is a vector composed of all the elements in T_i and Ψ is a diagonal matrix composed of all ψ_i. Similarly, we can reformulate our optimization problem (4.4) in terms of a linear system:

$$\min_{\tilde{v}} \left\|\widetilde{A}\tilde{v} - \tilde{t}\right\|_2^2 \tag{4.9}$$

where $\tilde{v}$ is the same as in (4.8), $\widetilde{A}$ and $\tilde{t}$ are derived from A and t of (4.8). Below we will show how to get (4.8) and extend it to (4.9). First, recall that

$$V_i = (v_4^i - v_1^i, v_4^i - v_2^i, v_4^i - v_3^i)$$
$$\widetilde{V_i} = \left(\tilde{v_4}^i - \tilde{v_1}^i, \tilde{v_4}^i - \tilde{v_2}^i, \tilde{v_4}^i - \tilde{v_3}^i\right)$$

Note that only $\widetilde{V_i}$ is unknown. Now assume:

$$v_1^i = \begin{bmatrix} x_1^i \\ y_1^i \\ z_1^i \end{bmatrix} \quad v_2^i = \begin{bmatrix} x_2^i \\ y_2^i \\ z_2^i \end{bmatrix} \quad v_3^i = \begin{bmatrix} x_3^i \\ y_3^i \\ z_3^i \end{bmatrix} \quad v_4^i = \begin{bmatrix} x_4^i \\ y_4^i \\ z_4^i \end{bmatrix}$$

$$\tilde{v_1}^i = \begin{bmatrix} \tilde{x_1}^i \\ \tilde{y_1}^i \\ \tilde{z_1}^i \end{bmatrix} \quad \tilde{v_2}^i = \begin{bmatrix} \tilde{x_2}^i \\ \tilde{y_2}^i \\ \tilde{z_2}^i \end{bmatrix} \quad \tilde{v_3}^i = \begin{bmatrix} \tilde{x_3}^i \\ \tilde{y_3}^i \\ \tilde{z_3}^i \end{bmatrix} \quad \tilde{v_4}^i = \begin{bmatrix} \tilde{x_4}^i \\ \tilde{y_4}^i \\ \tilde{z_4}^i \end{bmatrix}$$

then

$$V_i = \begin{bmatrix} x_4^i - x_1^i & x_4^i - x_2^i & x_4^i - x_3^i \\ y_4^i - y_1^i & y_4^i - y_2^i & y_4^i - y_3^i \\ z_4^i - z_1^i & z_4^i - z_2^i & z_4^i - z_3^i \end{bmatrix}$$

$$\widetilde{V_i} = \begin{bmatrix} \tilde{x_4}^i - \tilde{x_1}^i & \tilde{x_4}^i - \tilde{x_2}^i & \tilde{x_4}^i - \tilde{x_3}^i \\ \tilde{y_4}^i - \tilde{y_1}^i & \tilde{y_4}^i - \tilde{y_2}^i & \tilde{y_4}^i - \tilde{y_3}^i \\ \tilde{z_4}^i - \tilde{z_1}^i & \tilde{z_4}^i - \tilde{z_2}^i & \tilde{z_4}^i - \tilde{z_3}^i \end{bmatrix}$$

therefore, we can get

$$V_i^{-1} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

where $a, b, \dots, i$ correspond to the entries of the inverse matrix of V_i that can be evaluated from the coordinates of the v_1^i, v_2^i, v_3^i and v_4^i. Thus

$$\widetilde{V}_i V_i^{-1} - T_i = \begin{bmatrix} \tilde{x_4}^i - \tilde{x_1}^i & \tilde{x_4}^i - \tilde{x_2}^i & \tilde{x_4}^i - \tilde{x_3}^i \\ \tilde{y_4}^i - \tilde{y_1}^i & \tilde{y_4}^i - \tilde{y_2}^i & \tilde{y_4}^i - \tilde{y_3}^i \\ \tilde{z_4}^i - \tilde{z_1}^i & \tilde{z_4}^i - \tilde{z_2}^i & \tilde{z_4}^i - \tilde{z_3}^i \end{bmatrix} \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} - \begin{bmatrix} t_1^i & t_2^i & t_3^i \\ t_4^i & t_5^i & t_6^i \\ t_7^i & t_8^i & t_9^i \end{bmatrix} \quad (4.10)$$

and $\|\widetilde{V}_i V_i^{-1} - T_i\|_F^2$ is equivalent to

$$\left\| \begin{bmatrix} -a & 0 & 0 & -d & 0 & 0 & -g & 0 & 0 & a+d+g & 0 & 0 \\ \vdots & & & & & & \cdots & & & \vdots & & \\ \vdots & & & & & & \cdots & & & \vdots & & \\ 0 & 0 & -c & 0 & 0 & -f & 0 & 0 & -i & 0 & 0 & c+f+i \end{bmatrix} \begin{bmatrix} \tilde{x_1}^i \\ \tilde{y_1}^i \\ \tilde{z_1}^i \\ \tilde{x_2}^i \\ \tilde{y_2}^i \\ \tilde{z_2}^i \\ \vdots \\ \tilde{x_4}^i \\ \tilde{y_4}^i \\ \tilde{z_4}^i \end{bmatrix} - \begin{bmatrix} t_1^i \\ t_2^i \\ t_3^i \\ t_4^i \\ t_5^i \\ t_6^i \\ t_7^i \\ t_8^i \\ t_9^i \end{bmatrix} \right\|_2^2$$

From left to right we have a 9×12 sparse matrix, a 12×1 column vector containing $\tilde{v_1}^i$, $\tilde{v_2}^i$ $\tilde{v_3}^i$ and $\tilde{v_4}^i$, and a 9×1 column vector containing all elements of the T_i. Similarly, for a model with n faces, q anchors, and k vertices, we can construct a $9n \times 3(n+k)$ sparse matrix A, a $3(n+k) \times 1$ column vector $\tilde{v}$ containing all $\tilde{v}_i$ and $\tilde{v_4}^i$,

$$\tilde{v} = \begin{bmatrix} \tilde{x_1} & \tilde{y_1} & \tilde{z_1} & \dots & \tilde{x_k} & \tilde{y_k} & \tilde{z_k} & \tilde{x_4}^1 & \tilde{y_4}^1 & \tilde{z_4}^1 & \dots & \tilde{x_4}^n & \tilde{y_4}^n & \tilde{z_4}^n \end{bmatrix}^{\mathrm{T}}$$

and finally, a $9n \times 1$ column vector t containing all the elements of $T_1, T_2, \dots, T_n$

$$t = \begin{bmatrix} t_1^1 & \dots & t_9^1 & \dots & t_1^n & \dots & t_9^n \end{bmatrix}^{\mathrm{T}}$$

Note that for an anchor, $\widetilde{V}_i V_i^{-1} = T_i$, therefore,

$$\sum_i^{n-q} \psi_i \left\| \widetilde{V}_i V_i^{-1} - T_i \right\|_F^2$$

is equivalent to

$$\|\Psi\left(A\tilde{v}-t\right)\|_2^2$$

where Ψ is a diagonal matrix composed of all ψ_i. Hence (4.1) becomes the linear system (4.8). To get (4.9), we need to expand (4.8) as illustrated in Figure 4.1: where A' and C are to account for the position constraint term $\sum_j^m \beta\|\tilde{v_j}-c_j\|_2^2$ in

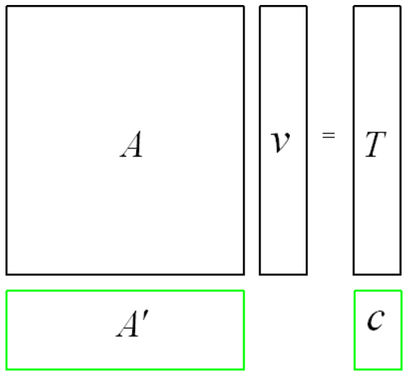


Figure 4.1: Expansion of linear system

(4.4), and then let

$$\widetilde{A}=\begin{bmatrix}\Psi' A\\ A'\end{bmatrix} \qquad \text{and} \qquad \tilde{t}=\begin{bmatrix}\Psi' T\\ C\end{bmatrix}$$

where Ψ' is a $9n\times 9n$ matrix based on Ψ of (4.8), but its diagonal values become $\sqrt{\alpha\psi_i}$ and each will be repeated 9 times on the diagonal.

$$\Psi'=\begin{bmatrix}\sqrt{\alpha\psi_1} & \dots & 0\\ \vdots & \dots & \vdots\\ 0 & \dots & \sqrt{\alpha\psi_{9n}}\end{bmatrix}$$

We then arrived at (4.9) which can be solved using the normal equation to obtain a least-squares solution.

Chapter 5

Results

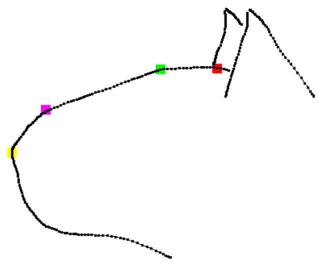
We have implemented the system using C++ and Graphite [8]. Graphite is a research platform for computer graphics, 3D modeling and numerical geometry. It is a cross-platform toolkit which works under both Windows XP and Linux.

5.1 Alignment

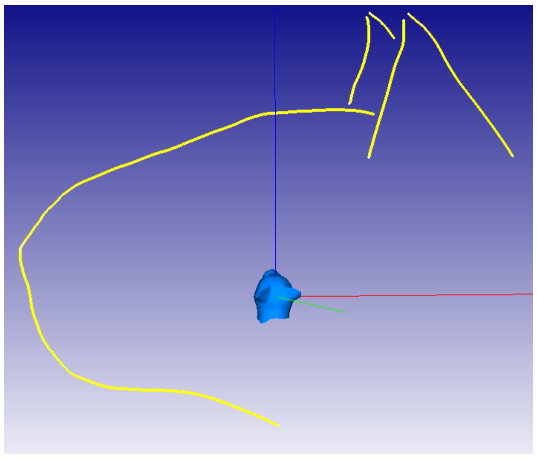
We use at least 4 correspondences between the sketch and the model. Additional correspondences do not necessarily better align the model with the sketch, instead that may introduce more errors. The alignment transformation matrices are obtained from solving the optimization problems described in Chapter 3, requiring a moderately good initial guess to avoid the local minima. The results are presented in Figure 5.1, 5.2, and 5.3.

5.2 Deformation

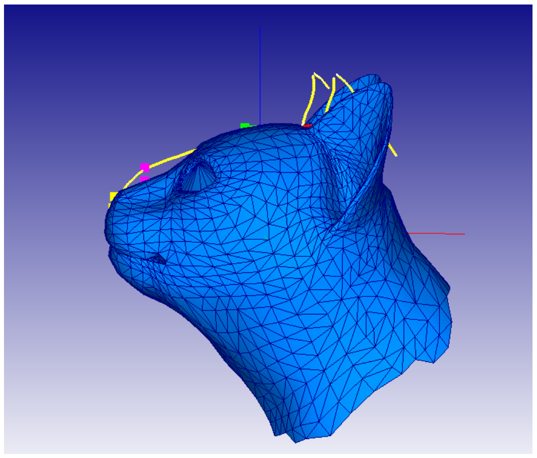
After the alignment, we may need to manually establish more correspondences other than those we used for the alignment to obtain a better match. Therefore, we use extra correspondences when necessary. Our system handles small deformations better than large deformations. The extent of deformations is determined by the extent of differences between normals and the positions of the correspondences. Large deformations imply a large change in orientation of normals and/or significant distance between corresponding sketch and model points. The results are shown in Figure (5.4, 5.5, 5.6).



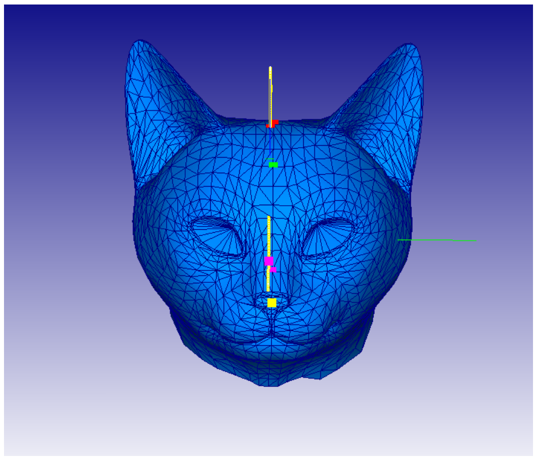
(a) 2D Sketch



(b) 2D sketch and 3D template before alignment

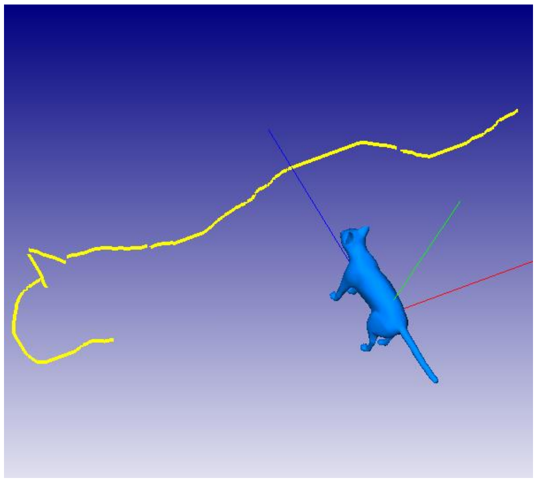


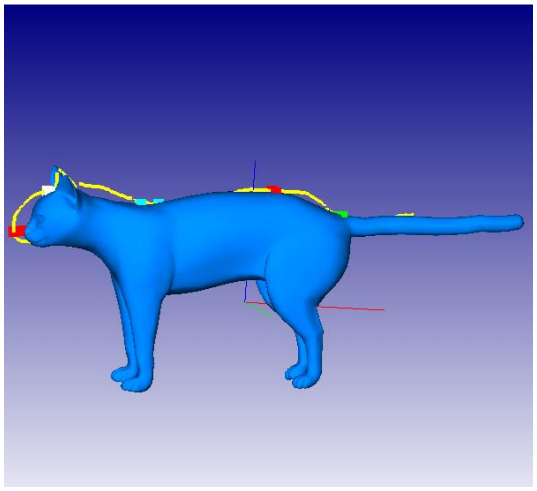
(c) After alignment, side view

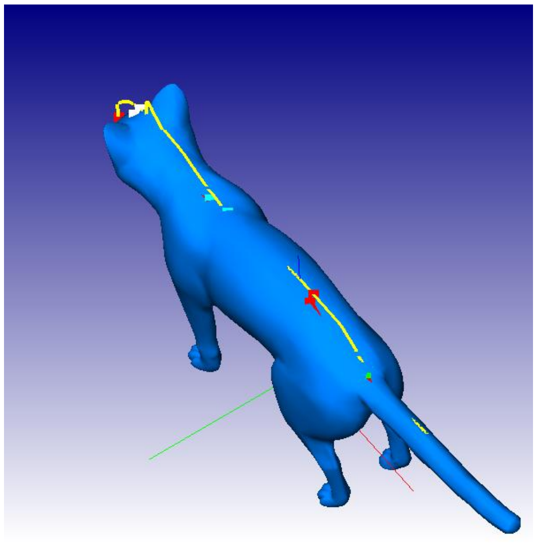


(d) After alignment, front view

Figure 5.1: Cat head example - The model has 3,748 triangles and 1,893 vertices, and is aligned n 0.5 second with 4 correspondences on forehead, face, nose and nose tip. The alignment


(a) 2D sketch with 3D cat template before alignment


(b) After alignment, side view


(c) After alignment, top left view

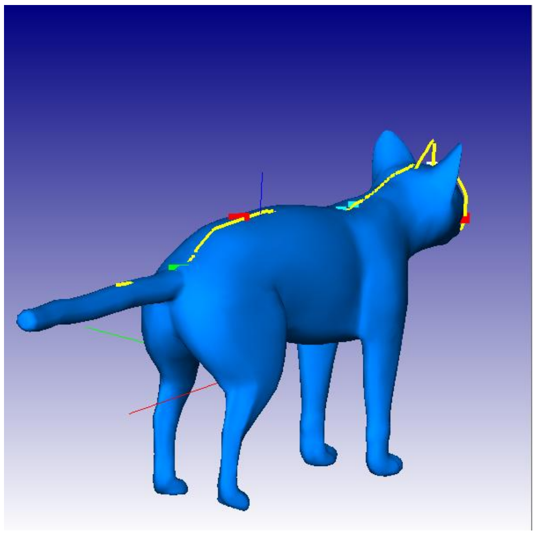
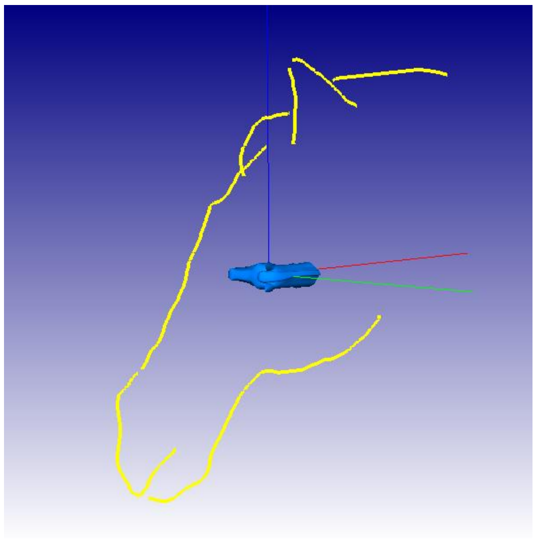
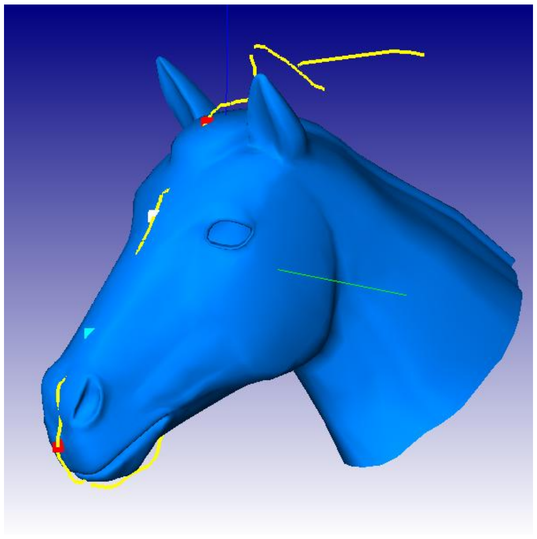

(d) After alignment, top right view

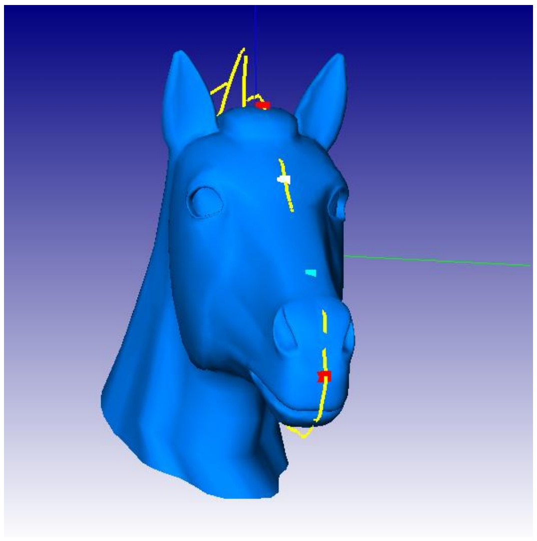
Figure 5.2: Cat example - The model has 14,410 triangles and 7,207 vertices, and is aligned 0.7 second with 5 correspondences. 2 on head, 1 on neck, 1 on back and 1 on rear



(a) 2D sketch with 3D horse head template before alignment

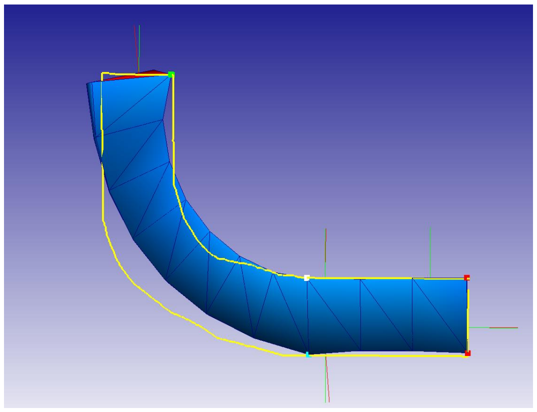


(b) After alignment, side view

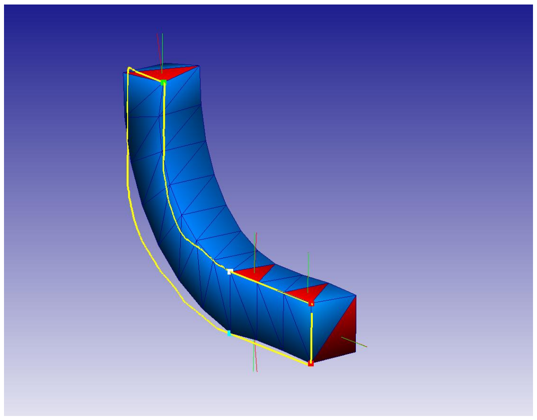


(c) After alignment, front view

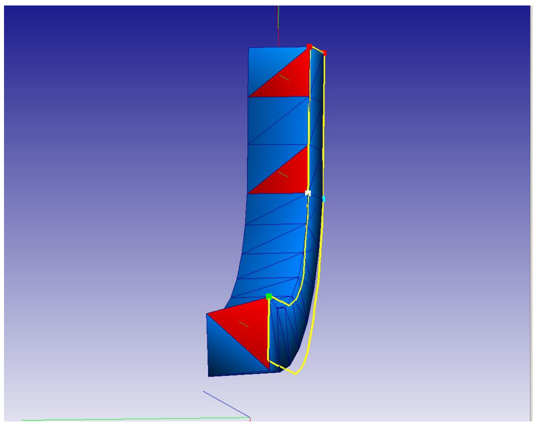
Figure 5.3: Horse head example - The model has 5,123 triangles and 2,589 vertices, and is aligned 0.5 second with 4 correspondence on forehead, face, nose and nose tip



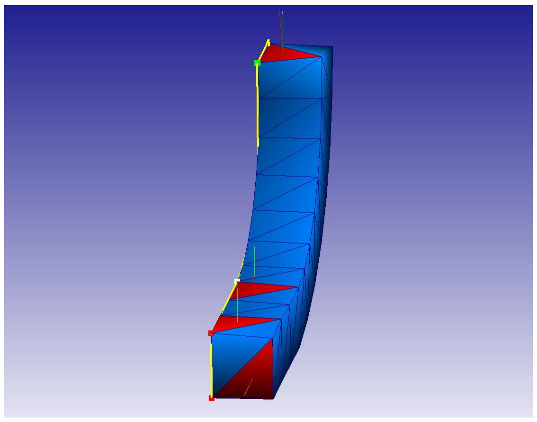
(a) Side view



(b) left side and front view

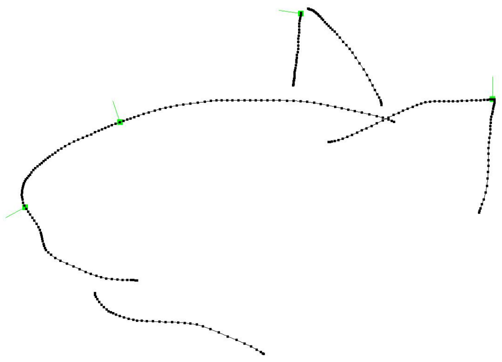


(c) Top view

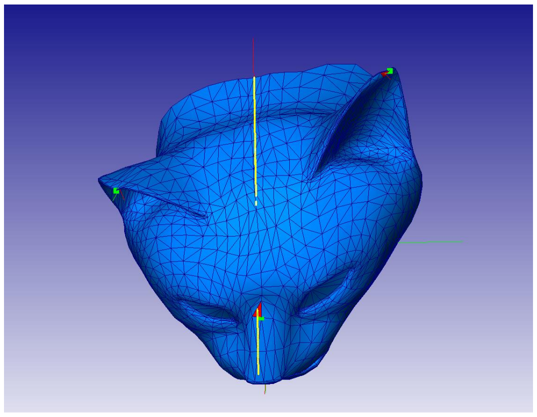


(d) Front view

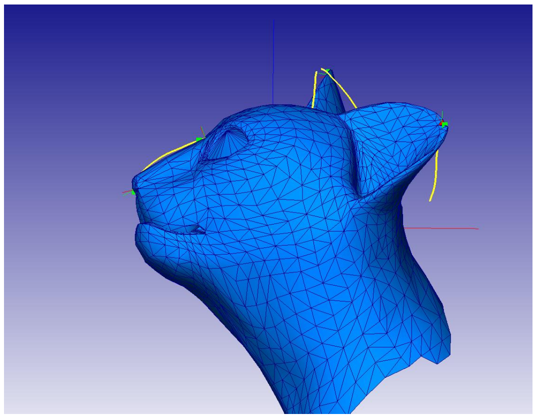
Figure 5.4: Simple deformation test with a bar - 4 correspondences for alignment at the horizontal part of the bar, see right part of the bar in (a). For deformation, based on existing 4 correspondences from alignment, we add one more on the left end of the bar (please refer to (a)), so use 5 correspondences to deform the model. We pick $\alpha = 0.2$ and $\beta = 1.0$. Please note that we highlight the anchor facets with red color.



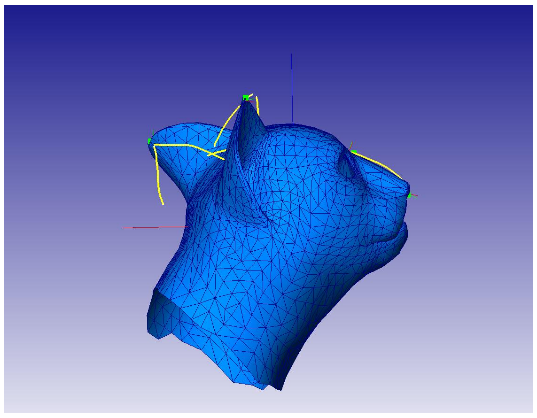
(a) 4 correspondences, 2 on ears and 2 on nose. and the positions of ear tips are deliberately set to one front and one back



(b) Top view

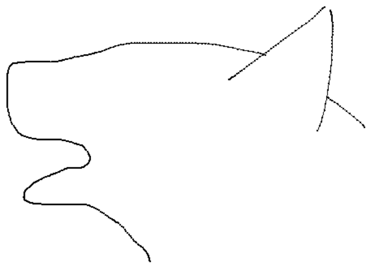


(c) left side view

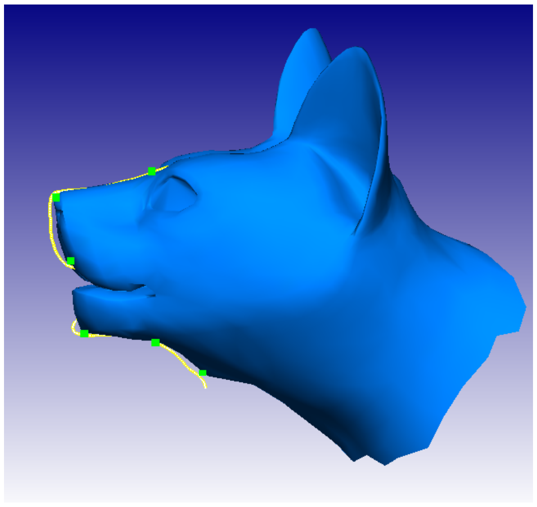


(d) Right side view

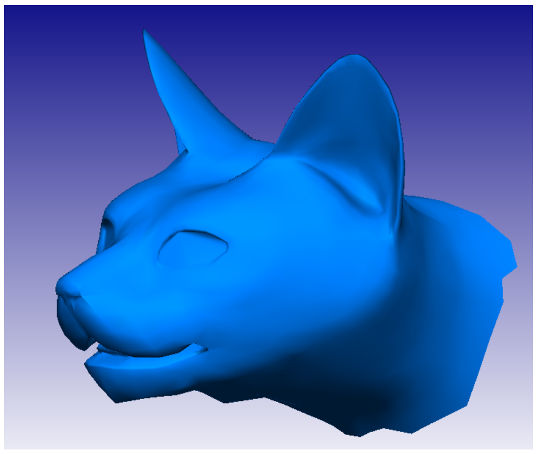
Figure 5.5: Cat head deformation test with 4 correspondences after alignment. $\alpha = 0.2$ and $\beta = 1.0$



(a) 2D sketch



(b) After alignment, use new correspondences to deform the model so that the shape match the sketch as closely as possible



(c) Final result

Figure 5.6: Cat head alignment and deformation example

Chapter 6

Conclusions and Future Work

The construction of 3D models from sketches is far from trivial. A reliable and capable system needs to interpret user drawings correctly and be able to generate a visually pleasing shape. Currently, our system uses simple sketches, such as the profile of a virtual 3D model; but we aim in the future to be able to use feature lines as well, like in [20] where they modify the features of a model.

The alignment of the 3D model to the 2D sketch plays an essential role in our system. It significantly affects the final result. However, in the alignment process, using penalty method with Newton iteration to solve a minimization problem with equality constraints is not guaranteed to give a meaningful result unless the initial guess is good. Our current system is problematic in this respect.

Deforming the 3D geometry to obtain a desired shape poses challenge with respect to the efficiency and quality of the results. From our experiments, it shows that the processing time significantly increases when the number of vertices gets large. The system does not work well with large deformations. Therefore, we need to improve from these two aspects. One way to mitigate the influence of large deformations is that we could use the concepts of the region of interests or the mesh material as proposed in [22], such that the deformation would not be propagated through out the entire mesh.

Deformation and vertex repositioning can potentially introduce bad shaped triangles in the resulting mesh. Therefore, after deformation step, we may need to remesh the new model. Any easy to use and robust method of remeshing would likely be suitable.

Bibliography

[1] M. Alexa. 2001. Local Control for Mesh Morphing. In Proceedings of Shape Modeling International, pages 209-215. IEEE Computer Society Press, 2001.

[2] M. Alexa. 2003. Differential coordinates for local mesh morphing and deformation. In *The Visual Computing 19, 2*, pp105-114, 2003

[3] M. Alexa, D. Cohen-Or, D. Levin. 2000. As-rigid-as possible shape interpolation. In *SIGGRAPH '00: ACM Trans. Graph.*, ACM Press, pp157-164, 2000.

[4] K.S. Arun, T.S. Huang, and S.D.Blostein 1987. Least square fitting of two 3-D point sets. In *IEEE Trans. Patt. Anal. Machine Intell.*vol. PAMI-9, no.5, pp 698-700, 1987.

[5] P. J. Besl, and N. D. McKey 1992. A Method for Registration of 3-D Shapes. In *IEEE Transactions on pattern analysis and machine intelligence*, Vol, 14, No.2, Feberuary 1992.

[6] M. Botsch, M. Pauly, C. Rossl, S. Bischoff, L. Kobbelt 2006. SIGGRAPH'06 Course on: Geometry modeling based on triangle meshes.

[7] D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, A. Santella. 2003. Suggestive Contours for Conveying Shape. In *SIGGRAPH '03: ACM Trans. Graph.*, ACM Press, pp848-855, 2003.

[8] Graphite 2003. http://www.loria.fr/ levy/Graphite/index.html, 2003

[9] B.K.P. Horn 1987. Closed-form solution of absolute orientation using unit quaternions. In *J.Opt. Soc. Amer. A*, vol. 4, no.4 pp629-642, Apr. 1987.

[10] J. Huang, X. Shi, X. Liu, K. Zhou, L. Wei, S. Teng, H. Bao, B. Guo, H. Shum. 2006. Subspace Gradient Domain Mesh Deformation. In *SIGGRAPH '06: ACM Trans. Graph.*, ACM Press, 2006.

[11] T. Igarashi, S. Matsuoka, and H. Tanaka 1999. Teddy: A Sketching Interface for 3D Freeform Design In *SIGGRAPH '99: ACM Trans. Graph.*, ACM Press, pp 409-416, 1999.

[12] T. Igarashi, T. Moscovich, and J. H. Houghes 2005. As-rigid-as possible shape manipulation In *SIGGRAPH '05: ACM Trans. Graph.*, ACM Press, pp 1134-1141, 2005.

[13] O. A. Karpenko, and J. F. Hughes 2006. SmoothSketch: 3D free-form shapes from complex sketches. In *SIGGRAPH '06: ACM Trans. Graph.*, vol 25, , no. 3, pp 589-598, 2006.

[14] Y. Kho and M. Garland 2005. Sketching mesh deformations. In *SI3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games.*, pp 147-154, ACM Press, NY. 2005.

[15] J. J. LaViola Jr., R. Davis, and T. Igarashi 2006. SIGGRAPH'06 Course on: An Introduction to Sketch-Based Interfaces.

[16] H. Lensch, W. Heidrich, and H.-P. Seidel 2001. A Silhouette-based Algorithm for Texture Registration and Stitching. In *Graphical Models*, July 2001, pp. 245-262

[17] X. Li and I. Guskov and J. Barhak 2006. Robust Alignment of Multi-view Range Data to CAD Model. In *SMI '06: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)*, IEEE Computer Society, Washington, DC, USA.

[18] Y. Lipman, O. Sorkine, D. Cohen-Or, D. Levin, C. Rossl,and H.-P. Seidel. 2004. Differential coordinates for interactive mesh editing. In Proceedings of Shape Modeling International, pages 181C190. IEEE Computer Society Press, 2004.

[19] Y. Lipman, O. Sorkine, D. Levin, and D. Cohen-Or. Linear 2005. Linear rotation-invariant coordinates for meshes. In Proceedings In *SIGGRAPH '05: ACM Trans. Graph.*, ACM Press, 2005.

[20] A. Nealen, O. Sorkine, M. Alexa and D. Cohen-Or 2005. A Sketch-Based Interface for Detail-Preserving Mesh Editing. In *SIGGRAPH '05: ACM Trans. on Computer graphics*, vol. 24, no. 3, pp 1142-1147, 2005.

[21] J. Nocedal and S. Wright 1999. Numerical Optimization. *Springer Verlag*, 1999

[22] T. Popa, D. Julius, and A. Sheffer 2006. Material Aware Mesh Deformations. In *SMI '06: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)*, IEEE Computer Society, Washington, DC, USA.

[23] S. Rusinkiewicz, M. Levoy. 2001. Efficient variants of the ICP algorithm. In *Third International Conference on 3D Digital Imaging and Modeling(3DIM).*, 2001

[24] D. Mumford. 1994. Elastica and computer vision. In *Algebraic Geometry and Its Applications.*, C. L. Bajaj, Ed. Springer-Verlag New York Inc. 1994

[25] A. Sheffer and V. Kraevoy. 2004. Pyramid coordinates for morphing and deformation. In 3DPVT, pages 68C75, 2004.

[26] K. Singh, and E. Fiume 1998. Wires: A geometry deformation technique. In *SIGGRAPH '98: ACM Trans. Graph.*, ACM Press, pp 405-414, 1998.

[27] O. Sorkine 2005. Laplacian mesh processing. In *Eurographics 2005—State of the Art Reports.*, The Eurographics Association, Dublin, Ireland, Eurographics, 53–70.

[28] O. Sorkine, Y. Lipman, D. Cohen-Or, M. Alexa, C. Rossl, and H.-P. Seidel 2004. Laplacian surface editing. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry processing.*, pp 179-188, 2004

[29] R. W. Sumner and J. Popovic 2004. Deformation transfer for triangle meshes. In *SIGGRAPH '04: ACM Trans. on Computer graphics*, vol. 23, no. 3, 2004.

[30] L. R. Williams 1991. Shading in Two Dimensions. In *Graphics Interface'91*, pages 143-151, 1991.

[31] L. R. Williams 1994. Perceptual Completion of Occluded Surfaces. PhD thesis, University of Massachusetts.

[32] L. R. Williams 1997. Topological reconstruction of a smooth manifold-solid from its occluding contour. In *Intl. Journal of Computer Vision 23*, 1, 93-C108, 1997.

[33] C. Yang, D. Sharon, and M. van de Panne 2005. Sketch-based Modeling of Parameterized Objects. In *2nd Eurographics Workshop on Sketch-Based Interfaces and Modeling*, Dublin, August 28-29, 2005.

[34] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, H. Shum. 2004. Mesh Editing with Poisson-Based Gradient Field Manipulation. In *SIGGRAPH '04: ACM Trans. Graph.*, ACM Press, pp644-651, 2004.

[35] K. Zhou, J. Huang, J. Snyder, X. Liu, H. Bao, B. Guo, H. Shum. 2005. Large Mesh Deformation Using the Volumetric Graph Laplacian. In *SIGGRAPH '05: ACM Trans. Graph.*, ACM Press, pp496-503, 2005.